

Distribution Statistics Usage in DB2

- Frequency Statistics are Collected and Stored in the DB2 System Catalog
 - COLGROUP() Specification of RUNSTATS Can Collect Column Frequencies
 - Any Quantity of Most Frequent and Least Frequent Values can be Collected
 - SYSCOLDIST and SYSCOLDISTSTATS Store the Values and Frequencies
- Frequency Statistics are Used by the DB2 Optimizer to Calculate Filter Factor
 - Influences
 - Table Access Path and Index Selection
 - Table Join Sequence
 - Frequency Statistics Apply to
 - Dynamic or Static SQL with Embedded Literals
 - Static SQL with Host Variables Bound with REOPT(ALWAYS)
 - Dynamic SQL with Parameter Markers Bound REOPT(ONCE) or REOPT(ALWAYS)
 - Static or Dynamic SQL with Host Variables That Cannot be NULL (No NULL Indicator)



© YL&A 1999-2006

Query Performance without Distribution Statistics

- This Query Suffers From Poor Performance

```

SELECT *
FROM PDB2.CONTACTNOTICE AS CN
INNER JOIN TDB2.KEY_NOTICE AS KN
ON CN.COD_CLIENT =KN.COD_CLIENT
AND CN.COD_GENERATION =KN.COD_GENERATION
AND CN.ID_CONTACTNOTICE=KN.ID_CONTACTNOTICE
INNER JOIN PDB2.CUST AS CU
ON KN.COD_CLIENT =CU.COD_CLIENT
AND KN.COD_GENERATION =CU.COD_GENERATION
AND KN.ID_CUST_JOB =CU.ID_CUST
INNER JOIN PDB2.CUST AS CU2
ON KN.COD_CLIENT =CU2.COD_CLIENT
AND KN.COD_GENERATION =CU2.COD_GENERATION
AND KN.ID_CUST_1 =CU2.ID_CUST
INNER JOIN TDB2.KEY_PERSON AS KP
ON CU2.COD_CLIENT =KP.COD_CLIENT
AND CU2.COD_GENERATION =KP.COD_GENERATION
AND CU2.ID_CUST =KP.ID_CUST
WHERE CN.DOM_NOTIFY = 'FICH'
AND CN.COD_GENERATION = 'MB'
AND CN.COD_CLIENT = '0450'
AND CN.DOM_STATUS_NOTICE = 'ERL'
AND CU2.DOM_PERSONGROUP = 'PR'
AND CU2.COD_LAND_DIVISION = 'AT'
AND CU2.DOM_STATUS = 'BEST'
AND CU2.DOM_CARE_STATE = 'S';
    
```

Query Response: 20 min.
Expected Response: <1 min.

DB2 has used column statistics to compute filter factors for predicates, and determined that table CU2 will return only 1 row, and has picked that table first in the join sequence.

Optimizer Estimates

Table	Estimate	% of cardf
CUST (CU)	58,039	0.2%
CUST (CU2)	1	0.000003%
CONTACTNOTICE	704	0.0009%
KEY_PERSON	19,619	0.03%
KEY_NOTICE	156,163	0.65%

*Example Courtesy of Terry Purcell, IBM



© YL&A 1999-2006

Query Performance without Distribution Statistics

- We Can Compare Actual Row Counts to the Optimizer Estimate
 - The Data is Skewed, but the Current Statistics Don't Reflect That
 - Example of a Count Query for Table CU2:

```
SELECT COUNT(*) = 267,011
FROM PDB2.CUST AS CU2
WHERE CU2.DOM_PERSONGROUP = 'PR'
AND CU2.COD_LAND_DIVISION = 'AT'
AND CU2.DOM_STATUS = 'BEST'
AND CU2.DOM_CARE_STATE = 'S'
AND CU2.COD_CLIENT = '0450'
AND CU2.COD_GENERATION = 'MB'
```

For Column COD_LAND_DIVISION:

```
SELECT COUNT(*) = 26,149,368
FROM PDB2.CUST AS CU2
WHERE CU2.COD_LAND_DIVISION = 'AT'
```

- Default filter factor is 1/colcardf or 1/461 = .22%.
- But, table cardf = 28,926,293 so 'AT' is on 90.4% of all rows!

- Comparison of all Counts and Estimates:

Table	From Count Queries		Optimizer Estimate	
	Count	% of cardf	Estimate	% of cardf
CUST (CU)	420,973	1.45%	58,039	0.2%
CUST (CU2)	267,011	0.92%	1	0.000003%
CONTACTNOTICE	1,472	0.002%	704	0.0009%
KEY_PERSON	20,114	0.03%	19,619	0.03%
KEY_NOTICE	156,347	0.65%	156,163	0.65%



© YL&A 1999-2006

Distribution Statistics in Action

- RUNSTATS is Used to Collect Distribution Statistics
 - Statistics Used for Dynamic Queries with Embedded Literals
 - Results in Filter Factor That More Accurately Reflects Reality

```
RUNSTATS TABLESPACE IDVKUNDA.IDVCUST
TABLE(PDB2.CUST)
COLGROUP(COD_CLIENT) FREQVAL COUNT 10
COLGROUP(DOM_STATUS) FREQVAL COUNT 10
COLGROUP(COD_LAND_DIVISION) FREQVAL COUNT 10
COLGROUP(DOM_PERSONGROUP) FREQVAL COUNT 10
COLGROUP(DOM_CARE_STATE) FREQVAL COUNT 10
```

CONTACTNOTICE
now first table
accessed. Query
response
improved from 20
min to 20 sec

- Collecting Distribution Statistics on all Tables Results in a new Table Access Sequence

Table	From Count Query		New Optimizer Estimate	
	Count	% of cardf	Estimate	% of cardf
CUST (CU)	420,973	1.45%	419,204	1.45%
CUST (CU2)	267,011	0.92%	240,891	0.91%
CONTACTNOTICE	1,472	0.002%	704	0.0009%
KEY_PERSON	20,114	0.03%	19,619	0.03%
KEY_NOTICE	156,347	0.65%	156,163	0.65%



© YL&A 1999-2006

V8 SQL Distribution Statistics Challenge

- Distribution statistics are only gathered for the most or least *n* occurring values
- Suppose we had gathered the top 5 values for a column such as this chart shows

Table CARDF is 10,865,917, column COLCARDF is 457

TBNAME	NAME	COLVALUE	CARDF
TABLE1	COL1	AT	1,250,807
TABLE1	COL1	RP	400,211
TABLE1	COL1	GH	375908
TABLE1	COL1	FF	303,515
TABLE1	COL1	XZ	293,011

DB2 knows that about 1,250,807 rows will be returned from this query

```
SELECT *
FROM TABLE1
WHERE COL1 = 'AT'
```

For this query DB2 assumes 18,236 rows

```
SELECT *
FROM TABLE1
WHERE COL1 = 'YA'
```

DB2 has to assume a uniform distribution of the rest of the values

$$((1-(2623452/10865917))/(457-5)) * 10865917$$


© YL&A 1999-2006

V9 Histogram Statistics and Exploitation

- New type of data distribution statistics
- Used to enhance predicate selectivity estimation for better access paths
 - Eliminates the guesswork for values not recorded via frequency distribution statistics
- Histogram statistics provides a way of summarizing data distribution on interval scale
 - Either discrete or continuous
 - Divides up the range of possible values in a dataset into quantiles
 - For which a set of statistics parameters are collected
- Based on equal-depth histogram statistics
 - Cuts the whole value range so that each quantile has about the same number of rows
 - Total number of quantiles
 - For each quantile
 - The pair of LOWVALUE/HIGHVALUE
 - Number of distinctive values (CARD)
 - Frequency (number of rows)
- Multi-column Histogram statistics are gathered the same as multi-column frequency statistics
 - Concatenated multiple columns and treat as a single value



© YL&A 1999-2006

V9 Histogram Statistics and Exploitation (cont.)

- **Distribution statistics are important for best query optimization**
 - DB2 chooses the best access path based upon the cost of predicate selectivity estimations and this relies on distribution statistics
- **Frequency statistics were for a single value**
 - Single-column or multi-column
 - Multi-column was a concatenation of multi-column values
 - Generally collected on most biased values
 - Most frequent or least frequent
 - In some cases this may not help DB2 with predicate selectivity other than uniform interpolation
- **With histogram statistics**
 - Get the distribution over the entire range of values
 - Predicate selectivity gets more accurate calculation
 - If searching range matches boundary of a quantile or group of consecutive quantiles
 - Interpolation is done in a much smaller granularity and predicate selectivity is evaluated with more accuracy



© YL&A 1999-2006

V9 Histogram Statistics

- **Histogram statistics are gathered for each quantile to a maximum of 100 quantiles**
- **This covers the entire range of values via RUNSTATS**

COLGROUP(COL1) HISTOGRAM NUMQUANTILES 5

Table CARDF is 10,865,917, column COLCARDF is 457

TBNAME	NAME	QUANTILENO	LOWVALUE	HIGHVALUE	CARDF	Freq
TABLE1	COL1	1	AA	AT	5	21%
TABLE1	COL1	2	AU	FF	118	19%
TABLE1	COL1	3	GA	GH	8	20%
TABLE1	COL1	4	GI	RP	212	20%
TABLE1	COL1	5	RQ	XZ	114	20%

DB2 assumes that no rows will be returned from this query

$(1/(\text{quantile cardf})) * (10865917 * (\text{quantile freq\%}))$

```
SELECT *
FROM TABLE1
WHERE COL1 = 'YA'
```

The entire range of values are represented

Assuming the COLCARDF of the quantile and frequency % of the quantile This is more accurate than with just the frequency distribution statistics



© YL&A 1999-2006

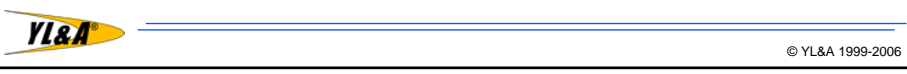
Optimistic Locking Challenge in V8

- Use optimistic locking to insure update integrity
 - All tables have an update timestamp column
 - Read data WITH UR
 - Include the timestamp in any UPDATE or DELETE
- Highly concurrent alternative to SELECT.....FOR UPDATE OF

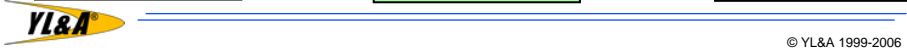
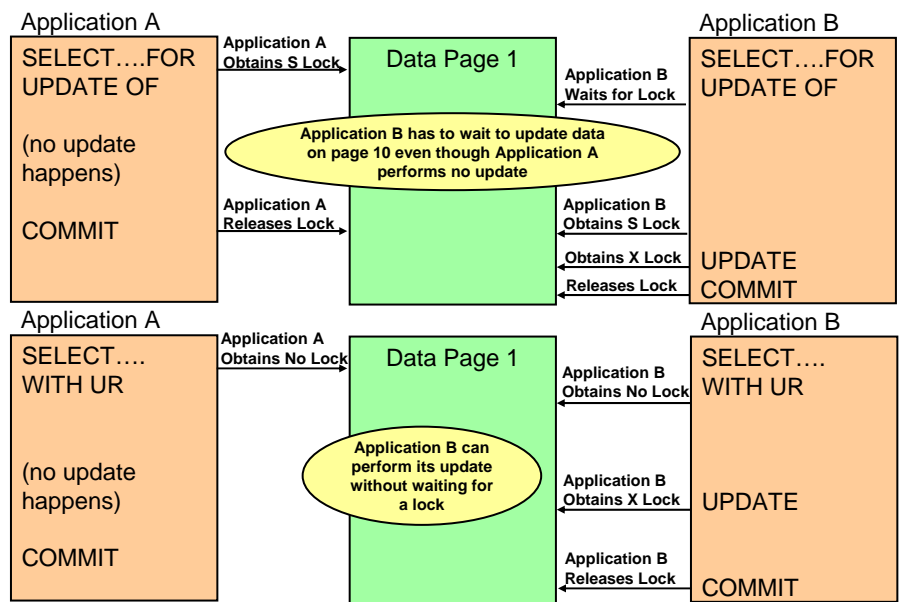
```
SELECT PKG_ID, PKG_DESC, UPD_TSP
FROM PKG_TBL
WHERE PKG_ID = :PKG-ID WITH UR
```

Optimistic locking uses the update timestamp to insure no other process has modified the data, and is built into DB2 for z/OS V9

```
UPDATE PKG_TBL
SET PKG_DESC = :PKG-DESC
,UPD_TSP = CURRENT TIMESTAMP
WHERE PKG_ID = :PKG_ID
AND UPD_TSP = :UPD-TSP
```



Optimistic Locking in Action



Optimistic Locking in V9

- **Optimistic Locking (a.k.a concurrency control)**
 - Faster more scalable locking alternative to database locking for concurrent access
 - Used to minimize the time for which a given resource is unavailable for use by other transactions
 - Locks for Reads
 - Obtained immediately before a read operation and released
 - Locks for Updates
 - Obtained immediately before an update operation and held until the end of the transaction
- **Optimistic locking test whether the underlying data source has changed by another transaction since the last read operation**
 - Prior to optimistic locking, all columns marked for update and their values in last read operation are added explicitly through a WHERE clause in the UPDATE statement
 - Therefore the UPDATE fails if the underlying column values have been changed



© YL&A 1999-2006

Optimistic Locking in V9 (cont..)

- **Provides an easier and more efficient way to detect change in a row**
 - Application does not need to know all old values marked for update
- **New expressions**
 - ROW CHANGED TOKEN
 - Returns a token that represents a relative point in the row modification
 - Application can compare the current ROW CHANGE TOKEN value of a row with the ROW CHANGE TOKEN value stored when the row was last fetched
 - This will determine if there was an update
 - GENERATED ALWAYS (DEFAULT) FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
 - Stored the time when the row is last changed (or initially insert)
 - Provides a way to capture the timestamp of the most recent change to a row
 - HIDDEN
 - Provided in column definition so timestamp does not show in *
 - RID
 - Built in function that returns a value that uniquely identifies a row



© YL&A 1999-2006

Optimistic Locking Programming in V9

- It is optimistic locking support
 - Not automatic optimistic locking
 - You still have to program for it
- Advantage is that DB2 is in control
 - At page level if no ROW CHANGE TIMESTAMP column
 - At row level if there is a ROW CHANGE TIMESTAMP column

```
SELECT PKG_ID, PKG_DESC,  
       ROW CHANGE TIMESTAMP FOR PKG_TBL  
FROM PKG_TBL  
WHERE PKG_ID = :PKG-ID WITH UR
```



```
UPDATE PKG_TBL  
SET PKG_DESC = :PKG-DESC  
WHERE PKG_ID = :PKG_ID  
AND ROW CHANGE TIMESTAMP FOR PKG_TBL = :UPD-TSP
```



© YL&A 1999-2006